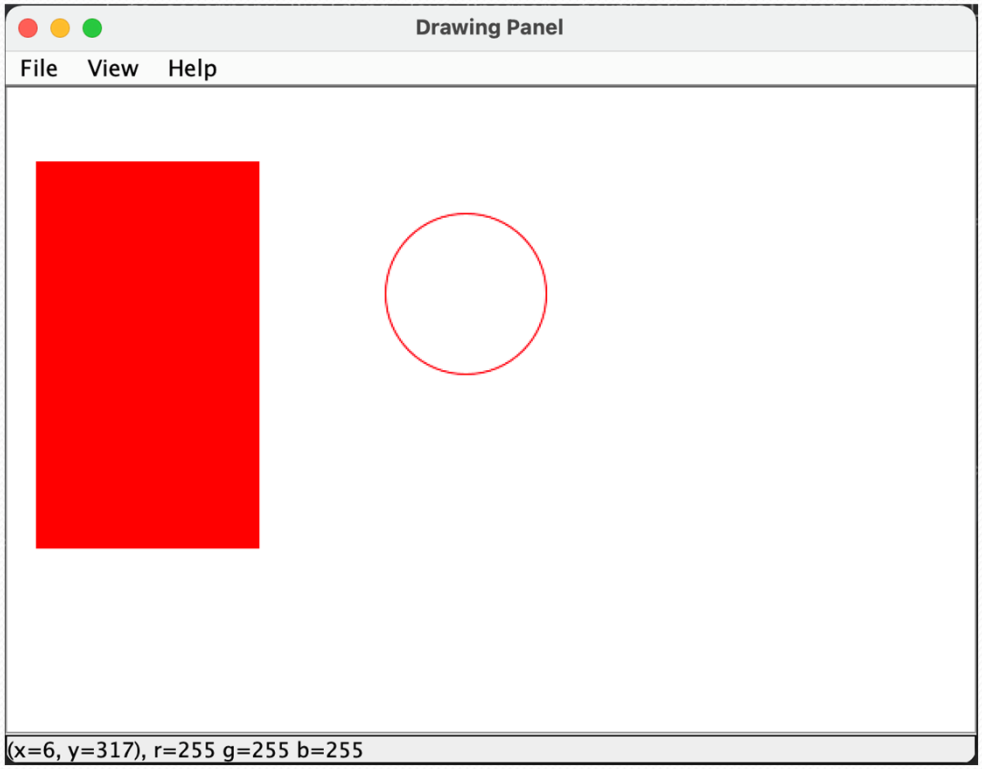
Graficación

Objetos gráficos

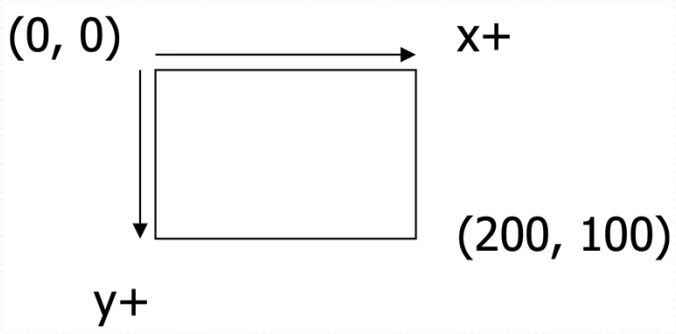
- Se dibujan gráficos usando 3 clases:
- DrawingPanel. Una ventana en la pantalla.
- No forma parte de Java; proporcionada por los autores del libro *Building Java Programs*. Ver la página del curso.
- Graphics2D. Una pluma para dibujar formas y líneas en un panel.
- Color. Colores con los cuáles se dibujan las formas y líneas.

DrawingPanel



Sistema de coordenadas

- Cada posición (x, y) es un *pixel* (picture element).
- El origen $(0, 0)$ está en la esquina superior izquierda de la ventana.
- El eje x se incrementa a la derecha.
- El eje y se incrementa hacia abajo.
- Una ventana de 200 x 100 pixeles se ve así:



DrawingPanel

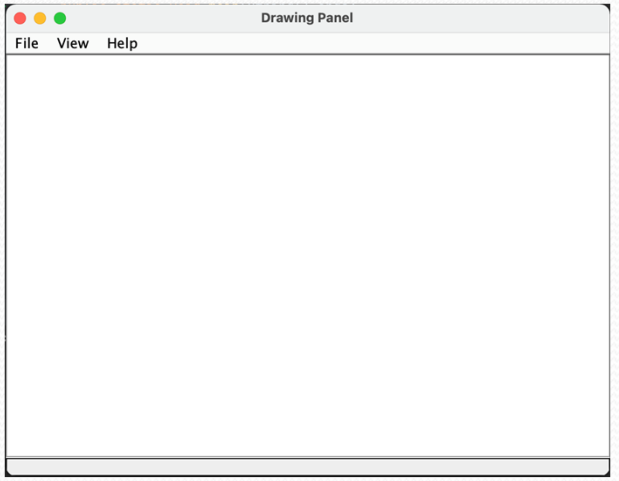
- Para crear una ventana:

  DrawingPanel name = new DrawingPanel(width, height);

- Ejemplo:

  DrawingPanel panel = new DrawingPanel(600, 400);

- La ventana está vacía.
- Se dibujan formas y líneas usando un objeto de tipo Graphics2D.



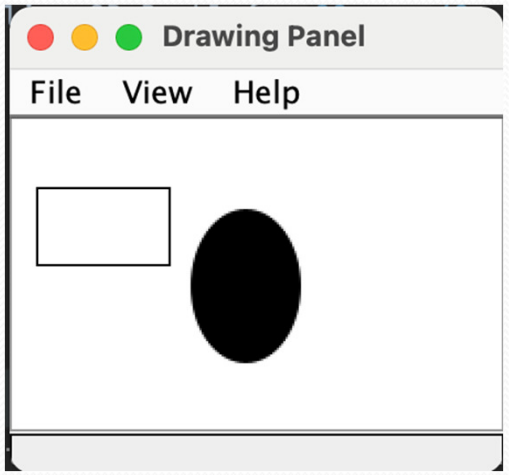
Graphics2D

- Ofrece métodos para dibujar líneas y formas.
- Se obtiene una referencia llamando a getGraphics en el DrawingPanel.

```
Graphics2D g = panel.getGraphics ();
```

- Se dibujan formas llamando al método draw o fill en el objeto Graphics2D.

```
g.draw(new Rectangle(10, 30, 60, 35));
g.fill(new Ellipse2D.Double(80, 40, 50, 70));
```



Graphics2D

- Graphics2D extiende a la clase Graphics.
- Para dibujar, también se pueden invocar los métodos de Graphics.

Algunos métodos de Graphics2D

addRenderingHints(Map<?,?> hints)	Establece los valores de preferencias para el rendering.
draw(Shape s)	Dibuja el objeto Shape indicado en modo hueco.
drawString(String str, int x, int y)	Escribe texto en la posición (x, y).
fill(Shape s)	Dibuja el objeto Shape indicado en modo lleno.
hit(Rectangle rect, Shape s, boolean onStroke)	Regresa true si el objeto Shape intersecta el rectángulo dado.
setBackground(Color color)	Establece el color del fondo.
setPaint(Paint paint)	Establece el atributo Paint.
setStroke(Stroke s)	Establece el atributo Stroke.
setFont(Font font)	Establece el tipo de letra (font).

Algunos métodos de Graphics

drawArc(argumentos)	Dibuja el contorno de un arco circular o elíptico.
drawImage(argumentos)	Dibuja una imagen.
drawLine(int x1, int y1, int x2, int y2)	Dibuja una línea entre los puntos (x1, y1) y (x2, y2).
drawOval(argumentos)	Dibuja el contorno de un óvalo.
drawPolygon(Polygon p)	Dibuja el contorno de un polígono.
drawRect(int x1, int y1, int w, int h)	Dibuja el contorno de un rectángulo..
fillArc(argumentos)	Dibuja un arco circular o elíptico lleno.
fillOval(argumentos)	Dibuja un óvalo lleno.
fillPolygon(Polygon p)	Dibuja un polígono lleno.
fillRect(int x1, int y1, int w, int h)	Dibuja un rectángulo lleno.

Preferencias de rendering

- Definidas en la clase RenderingHints.
- El primero paso es obtener un objeto de tipo RenderingHints.
- El segundo paso es añadir las preferencias usando el método put.
- El tercer paso es agregar las preferencias usando el método addRenderingHints.
- Ejemplo:

Preferencias de rendering

```
RenderingHints hints = g.getRenderingHints();
hints.put(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
hints.put(RenderingHints.KEY_RENDERING,
RenderingHints.VALUE_RENDER_QUALITY);
hints.put(RenderingHints.KEY_TEXT_ANTIALIASING,
RenderingHints.VALUE_TEXT_ANTIALIAS_ON);
g.addRenderingHints(hints);
```

Shape

- La interface Shape define objetos que tienen una forma geométrica.
- La mayoría de las clases tienen dos versiones: Double y Float.

Arc2D	Rectangle
CubicCurve2D	Rectangle2D
Ellipse2D	RectangularShape
GeneralPath	RoundRectangle2D
Line2D	
Path2D	
Polygon	
QuadCurve2D	

Colores

- Hay colores predefinidos mediante constantes en la clase Color:

  Color.nombre-color

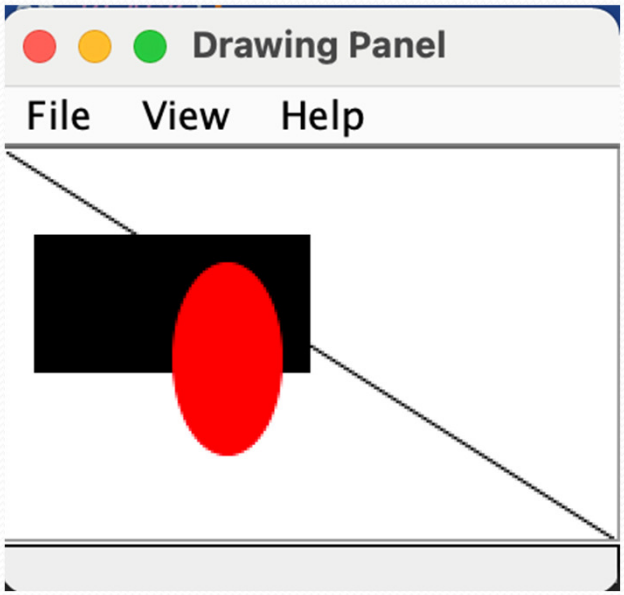
- Dónde nombre-color puede ser black, blue, cyan, darkGray, gray, green, lightGray, magenta, orange, pink, red, white, yellow.
- También se pueden usar mayúsculas, como BLACK, BLUE, etc.
- Se pueden crear colores usando valores RGB (Red-Green-Blue) entre 0 y 255.
- Por ejemplo:

```
Color brown = new Color(192, 128, 64);
```

Usando colores

- Se le pasa un Color al método setPaint de Graphics2D.
- Las siguientes formas se dibujan en ese color.

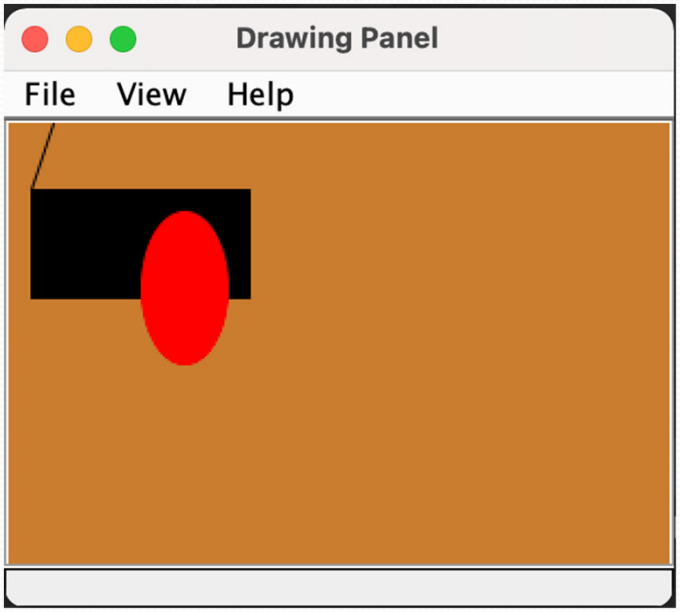
```
g.setPaint(Color.black);
g.fillRect(10, 30, 100, 50);
g.drawLine(0, 0, 220, 140);
g.setPaint(Color.red);
g.fillOval(60, 40, 40, 70);
```



Usando colores

- Para cambiar el color del fondo se le pasa un Color al método setBackground de DrawingPanel.

```
Color brown = new Color(192, 128, 64);
panel.setBackground(brown);
```



Stroke

- La interface Stroke permite decorar el contorno de un objeto Shape.
- La utiliza el método draw de Graphics2D.
- El método setStroke() se puede utilizar para:
  - Cambiar el ancho de las líneas y contornos.
  - Definir patrones para líneas y contornos.

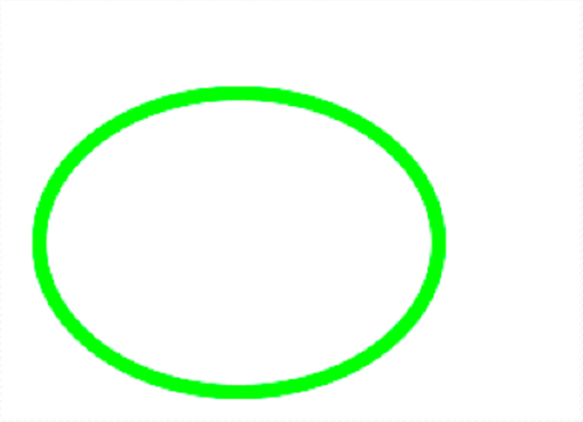
BasicStroke

- Es la única clase conocida que implementa Stroke.
- Constructor simple:

  BasicStroke(float width)
- width es el ancho en pixeles del contorno.

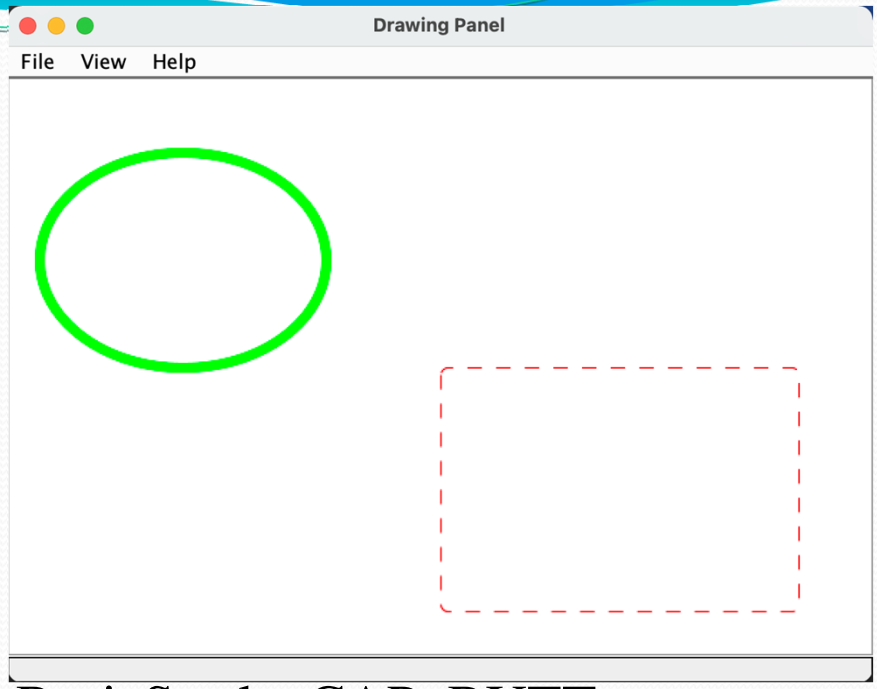
BasicStroke

```
g.setPaint(Color.green);
g.setStroke(new BasicStroke(7));
g.drawOval(20, 50, 200, 150);
```



Ejemplo

```
g.setColor(Color.green);
g.setStroke(new BasicStroke(7));
g.drawOval(20, 50, 200, 150);
g.setPaint(Color.red);
// 10 pixeles opacos y 10 transparentes
BasicStroke dashed = new BasicStroke(1.0f, BasicStroke.CAP_BUTT,
BasicStroke.JOIN_MITER, 10.0f, new float[] {10f}, 0.0f);
g.setStroke(dashed);
g.drawRoundRect(300, 200, 250, 170, 10, 10);
```



BasicStroke

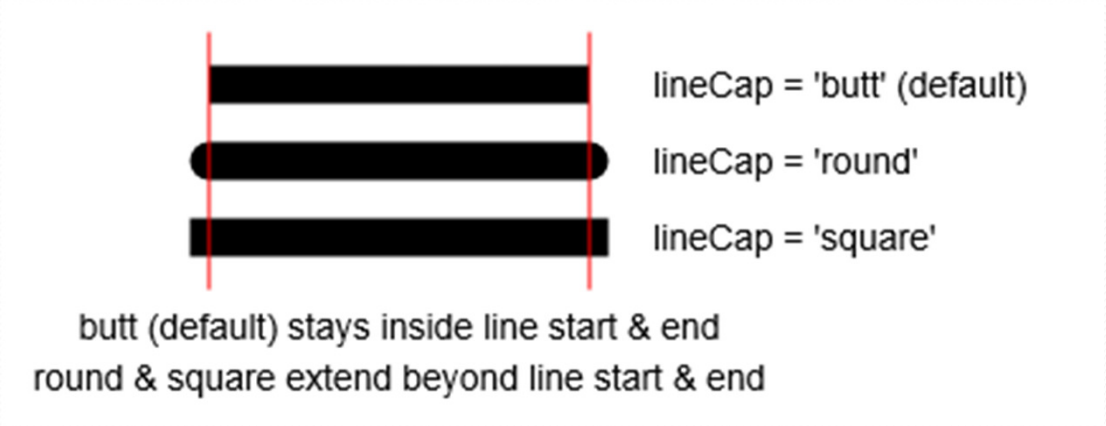
- Constructor más general:

  BasicStroke(float width, int cap, int join, float miterlimit, float[] dash, float dash_phase)

- Permite dibujar líneas o contornos segmentados.
- width es el ancho de la línea o del contorno.

BasicStroke

- cap es la decoración al final de cada segmento.
- Opciones: CAP_BUTT, CAP_ROUND, CAP_SQUARE.



Fuente: https://riptutorial.com/html5-canvas/example/13469/linecap--a-path-styling-attribute-

BasicStroke

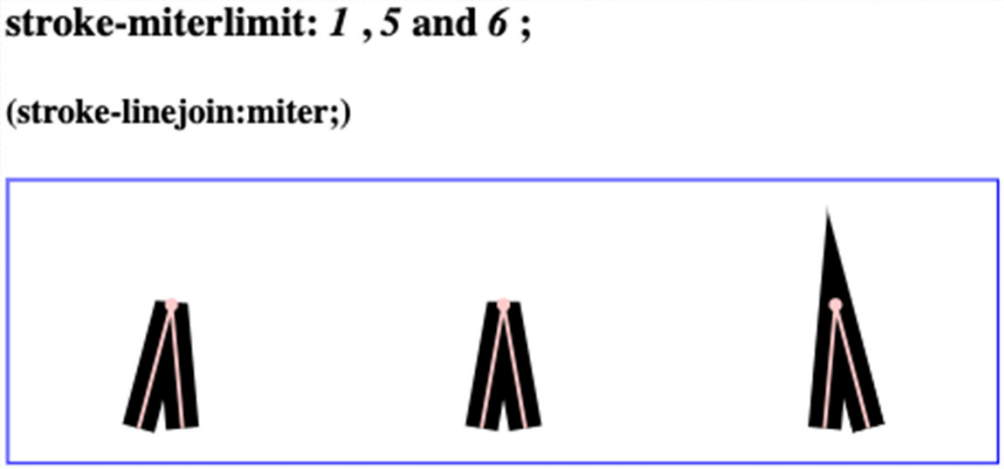
- join es la decoración en la unión de segmentos.
- Opciones: JOIN_ROUND, JOIN_BEVEL, JOIN_MITER.

stroke-linejoin: ***miter*** **|** ***round*** **|** ***bevel*****;**

Fuente: https://i.stack.imgur.com/og3Ya.png

BasicStroke

- miterlimit es el límite para recortar la unión (join) miter.
- Debe ser mayor o igual a 1.



Fuente: https://i.stack.imgur.com/og3Ya.png

BasicStroke

- dash es un arreglo que representa el patrón de punteado.
- El arreglo contiene la definición del patrón alternando secciones opacas y transparentes.
- Ejemplo:

```
float[] dash = new float[] {4, 2, 8, 2};
```

- El patrón es 4 pixeles opacos, 2 transparentes, 8 opacos, 2 transparentes y se repite.
- Si el arreglo tiene un solo elemento:

```
float[] dash = new float[] {10};
```

- El patrón es 10 pixeles opacos, 10 transparentes y se repite.

BasicStroke

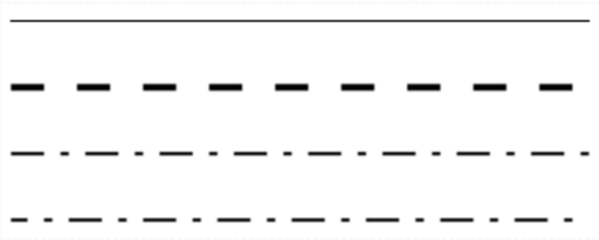
- dash_phase es el offset para comenzar el patrón de punteado (por ejemplo 0).

Ejemplo

```
g.setPaint(Color.red);
BasicStroke dashed = new BasicStroke(1.0f, BasicStroke.CAP_BUTT,
BasicStroke.JOIN_MITER, 10.0f, new float[] {10f}, 0.0f);
g.setStroke(dashed);
g.drawRoundRect(300, 200, 250, 170, 10, 10);
```

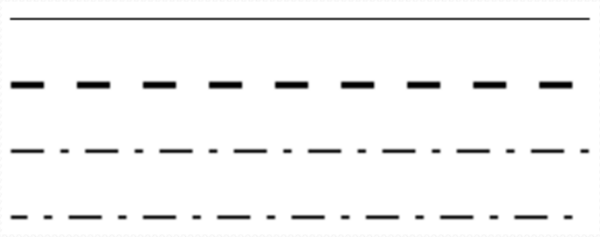
Ejemplo

```
g.drawLine(50, 40, 400, 40);     // Línea continua
BasicStroke stroke1 = new BasicStroke(4,   // ancho 4 pixeles
        BasicStroke.CAP_BUTT, BasicStroke.JOIN_ROUND, 1,
        new float[] {20, 20},        // 20 pixeles opacos, 20 transparentes
        0);
g.setStroke(stroke1);
g.drawLine(50, 80, 400, 80);
```



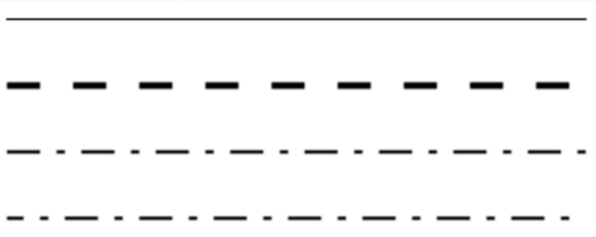
Ejemplo

```
BasicStroke stroke2 = new BasicStroke(2,    // ancho 2 pixeles
        BasicStroke.CAP_BUTT, BasicStroke.JOIN_ROUND, 1,
        // 20 pixeles opacos, 10 transparentes, 5 opacos, 10 transparentes
        new float[] {20, 10, 5, 10},
        0);
g.setStroke(stroke2);
g.drawLine(50, 120, 400, 120);
```



Ejemplo

```
BasicStroke stroke3 = new BasicStroke(2,    // ancho 2 pixeles
        BasicStroke.CAP_BUTT, BasicStroke.JOIN_ROUND, 1,
        // 20 pixeles opacos, 10 transparentes, 5 opacos, 10 transparentes
        new float[] {20, 10, 5, 10},
        10);    // dash phase = 10
g.setStroke(stroke3);
g.drawLine(50, 160, 400, 160);
```

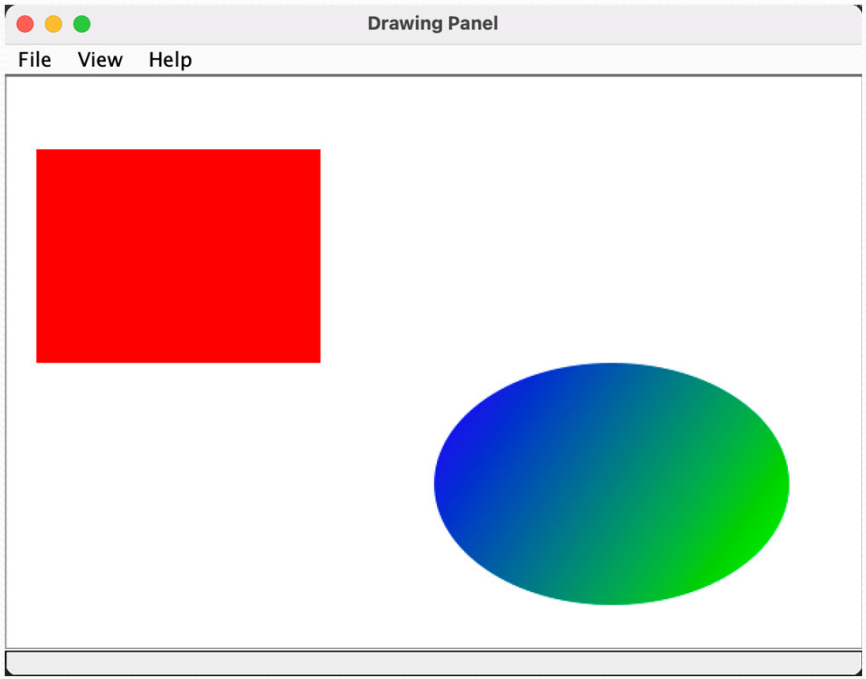


Paint

- La interface Paint define cómo se pueden generar patrones de color para las operaciones de Graphics2D.
- El método setPaint() puede usarse para establecer un color plano, un color en gradiente o para utilizar una imagen como textura.
- Ejemplos:

Ejemplo 1

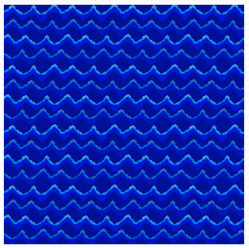
```
g.setPaint(Color.red);
g.fillRect(20, 50, 200, 150);
Paint paint = new GradientPaint(
                    300f, 200f, Color.blue,
                    550f, 370f, Color.green);
g.setPaint(paint);
g.fillOval(300, 200, 250, 170);
```

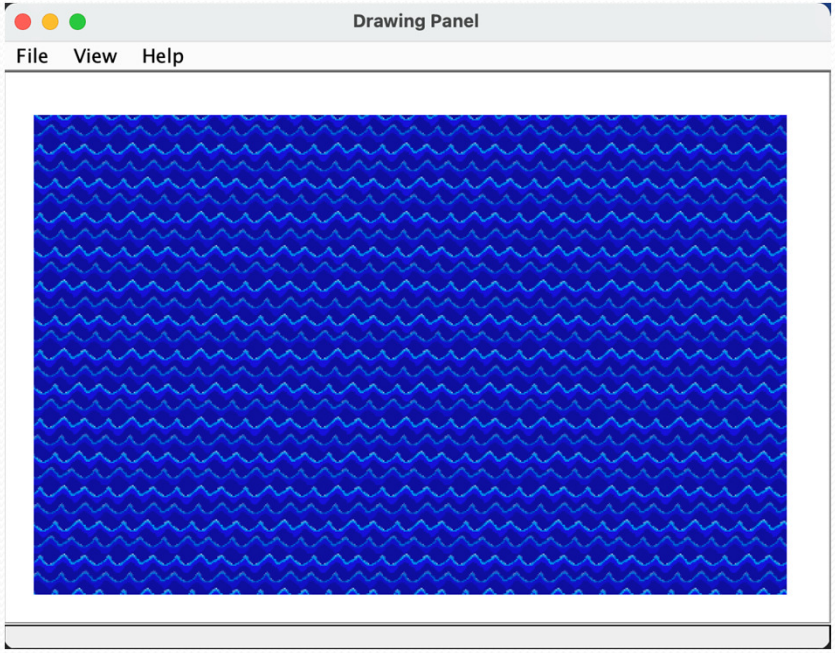


Ejemplo 2

```
BufferedImage image = ImageIO.read(new File("water.png"));
Paint paint = new TexturePaint(
        image, new Rectangle(0, 0, 200, 200));
g.setPaint(paint);
g.fillRect(20, 30, 550, 350);
```

water.png



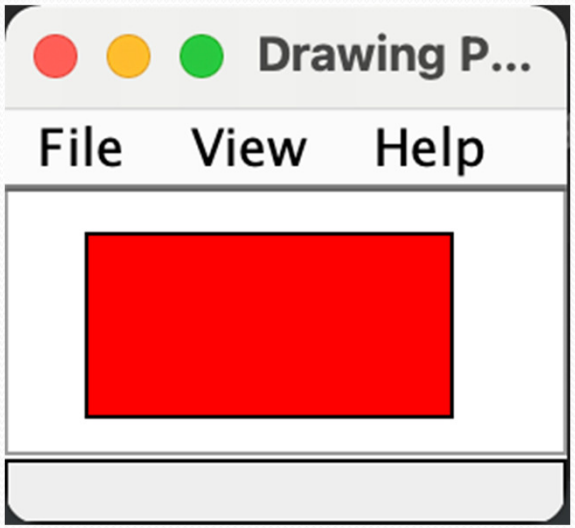


Formas con bordes

- Para dibujar una forma llena con borde de otro color, primero se dibuja rellena (*fill*) y luego se dibuja hueca (*draw*) con el color del borde deseado.

Formas con bordes

```
Rectangle2D r = new Rectangle2D.Double(20, 10, 100, 50);
g.setPaint(Color.red);      // relleno rojo
g.fill(r);
g.setPaint(Color.black);  // borde negro
g.draw(r);
```

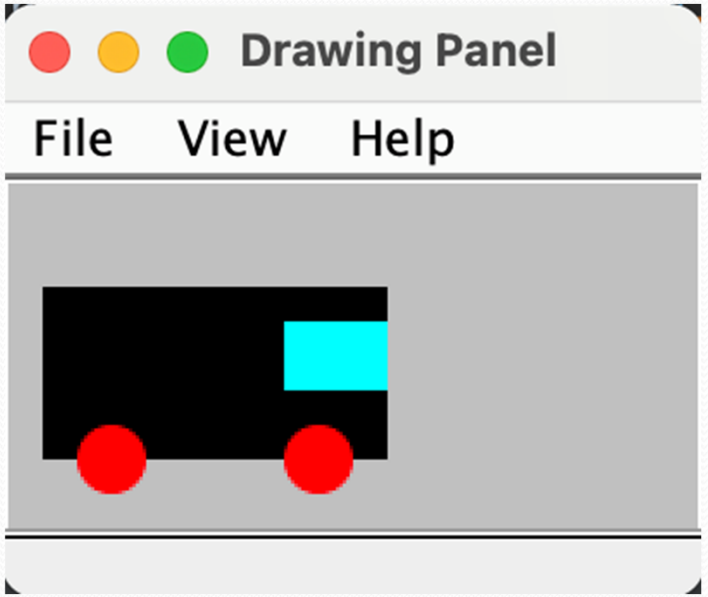


Sobreponer formas

- Si dos o más figuras ocupan los mismos pixeles, la última es la que gana.

Sobreponer formas

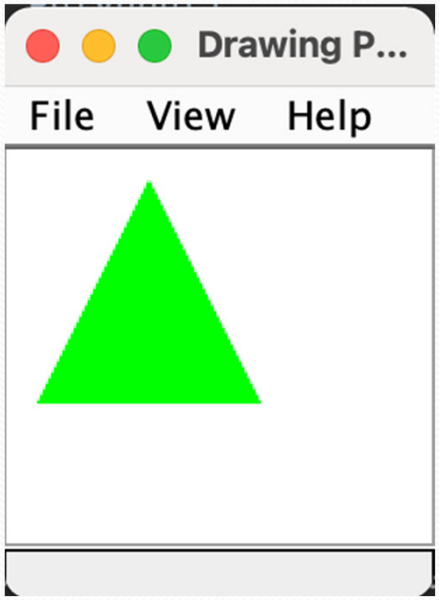
```
DrawingPanel panel = new DrawingPanel(200, 100);
Graphics g = panel.getGraphics();
panel.setBackground(Color.lightGray);
g.setPaint(Color.black);
g.fillRect(10, 30, 100, 50);
g.setPaint(Color.red);
g.fillOval(20, 70, 20, 20);
g.fillOval(80, 70, 20, 20);
g.setPaint(Color.cyan);
g.fillRect(80, 40, 30, 20);
```



Clase Polygon

- Objetos para representar formas arbitrarias.
- Se le agregan puntos a un Polygon usando su método addPoint(x, y).

```
DrawingPanel p = new DrawingPanel(100, 100);
Graphics2D g = p.getGraphics();
g.setColor(Color.green);
Polygon poly = new Polygon();
poly.addPoint(10, 90);
poly.addPoint(50, 10);
poly.addPoint(90, 90);
g.fillPolygon(poly);
```



Métodos de DrawingPanel

clear()	Borra las formas que se han dibujado en este panel
loadImage(nombre)	Carga la imagen almacenada en el archivo indicado
setWidth(width)	Cambia el ancho de este panel
setHeight(height)	Cambia el alto de este panel
setSize(width, height)	Cambia el tamaño de este panel
save(filename)	Guarda la imagen de este panel en un archivo png
sleep(ms)	Pausa el dibujo en el número dado de milisegundos

Font

- El tipo de letra se especifica usando el método setFont(font) sobre un objeto de tipo Graphics2D.
- La clase Font permite definir el tipo de letra para escribir texto.
- Constructor de Font:

  Font font = new Font(nombre, estilo, tamaño);

- Nombre es un String con el nombre del tipo de letra, por ejemplo: Monospaced, Serif, SansSerif, Arial, Helvetica, Times New Roman, etc.

Font

- Los tipos de letra disponibles en un sistema se pueden obtener así:

```
String[] fonts =
GraphicsEnvironment.getLocalGraphicsEnvironment().getAvailableFontFamily
Names();
```

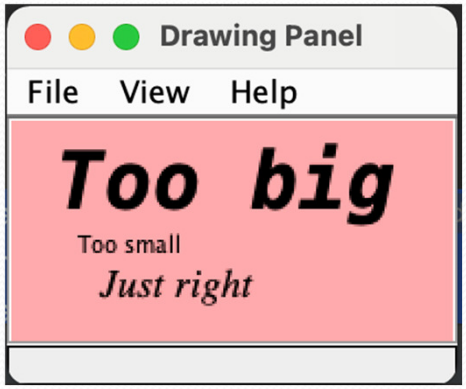
- Estilo es una constante de las definidas en la clase Font:

Font.BOLD	**Texto bold**
Font.ITALIC	*Texto italic*
Font.PLAIN	Texto plain
Font.BOLD + Font.ITALIC	***Texto bold e italic***

Font

- Tamaño es un entero que indica el tamaño del tipo de letra en *puntos tipográficos* (1 punto es aproximadamente 1/72 de pulgada).
- Para mayores detalles sobre el tamaño leer la descripción del método getSize() en el manual de la clase Font.

Ejemplo




```
DrawingPanel panel = new DrawingPanel(200, 100);
panel.setBackground(Color.PINK);
Graphics2D g = panel.getGraphics();
g.setFont(new Font("Monospaced", Font.BOLD + Font.ITALIC, 36));
g.drawString("Too big", 20, 40);
g.setFont(new Font("SansSerif", Font.PLAIN, 10));
g.drawString("Too small", 30, 60);
g.setFont(new Font("Serif", Font.ITALIC, 18));
g.drawString("Just right", 40, 80);
```

Imágenes

- DrawingPanel puede desplegar imágenes en formatos JPEG, PNG y GIF.
- Esto se hace en dos pasos:
- Paso 1: declarar un objeto de tipo Image y asignarle lo que regrese el método loadImage sobre un objeto de tipo DrawingPanel.
- Por ejemplo:

```
Image imagen = panel.loadImage("paisaje.jpg");
```

- Paso 2: desplegar la imagen llamando al método drawImage sobre un objeto de tipo Graphics2D.

Imágenes

- Por ejemplo, desplegar la imagen con su esquina superior izquierda en (20, 40):

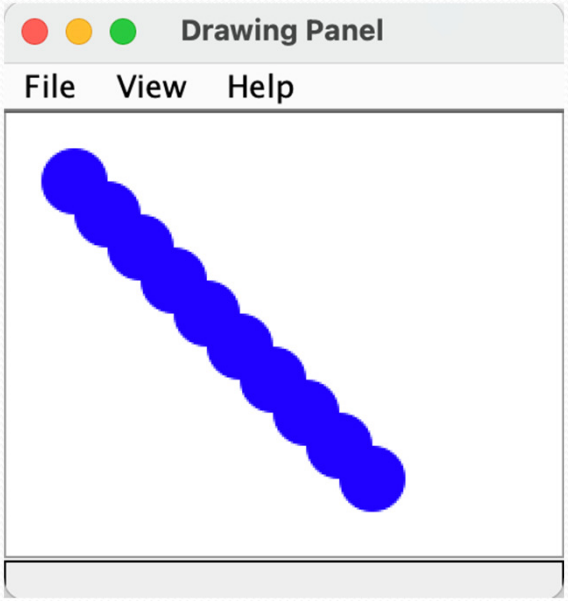
```
g.drawImage(imagen, 20, 40, panel);
```

Ejemplo

```
DrawingPanel panel = new DrawingPanel(190, 200);
Image imagen = panel.loadImage("triangle.png");
Graphics g = panel.getGraphics();
g.drawImage(imagen, 20, 40, panel);
g.setFont(new Font("Helvetica", Font.PLAIN, 20));
g.drawString("Un triángulo verde", 10, 180);
```

Animación con sleep

```
DrawingPanel panel = new DrawingPanel(250, 200);
Graphics2D g = panel.getGraphics();
g.setPaint(Color.blue);
for (int i = 1; i <= 10; i++) {
    g.fillOval(15 * i, 15 * i, 30, 30);
    panel.sleep(500);
}
```



Conclusión

- El código fuente DrawingPanel está en https://www.buildingjavaprograms.com/code-files/5ed/DrawingPanel.java
- El manual está en https://www.buildingjavaprograms.com/code-files/5ed/javadoc/DrawingPanel.html